CALIFORNIA INSTITUTE OF TECHNOLOGY

EARTHQUAKE ENGINEERING RESEARCH LABORATORY

PROCESSING OF NEAR-FIELD EARTHQUAKE ACCELEROGRAMS

BY

LUO-JIA WANG

REPORT NO. EERL 96-04

PASADENA, CALIFORNIA

SEPTEMBER 1996

THIS INVESTIGATION WAS SUPPORTED IN PART BY GRANTS CMS-9223680 AND CMS-9509877 FROM THE NATIONAL SCIENCE FOUNDATION, AND WAS CONDUCTED UNDER THE SUPERVISION OF PROFESSOR W. D. IWAN. ANY OPINIONS, FINDINGS, CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THE PUBLICATION ARE THOSE OF THE AUTHOR AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE NATIONAL SCIENCE FOUNDATION.

PROCESSING OF NEAR-FIELD EARTHQUAKE ACCELEROGRAMS

by

Luo-Jia Wang

Report No. EERL 96–04

A Report on Research Conducted under a Grant
from the National Science Foundation

Pasadena, California
September 1996

PROCESSING OF NEAR-FIELD EARTHQUAKE ACCELEROGRAMS

Luo-Jia Wang

Earthquake Engineering Research Laboratory
California Institute of Technology

The Near-Field pulse-like velocity and displacement time histories associated with a strong earthquake can greatly affect a wide range of different types of structures [1]. The important long period components in NEAR-FIELD earthquake accelerograms, however, are normally eliminated or distorted by conventional data processing which is based on band-pass filtering; for example, using an Ormsby filter with a low frequency cut-off of 0.2-0.4 Hz.

A new data processing technique has been proposed by Iwan and Chen [2] to recover the long period components from NEAR-FIELD earthquake accelerograms. This technique is based on the inverse of the data recording and retrieving procedure, which includes appropriate instrument correction according to the instrument type, and baseline correction without band-pass filtering. It may be briefly summarized by the following steps:

1. Apply a least-mean-square linear fit to the uncorrected accelerograms to eliminate any uncertainty in the instrument centering.

2. Apply instrument correction to compensate for the fact that the transfer function of the transducer is not flat over the entire frequency band.

3. Integrate the instrument corrected acceleration time history to obtain a raw velocity time history, assuming zero initial velocity. A trapezoidal integration rule and a central difference differentiation scheme are used herein.

4. Apply a segmented polynomial baseline fit to the raw velocity time history to remove any non-physical trends. Since the ground velocity physically begins at zero and ends at zero, the baseline is fitted to the initial and final portions of the raw velocity time history. These two polynomials are connected by the lowest order (smoothest) polynomial baseline connection that continuously connects the initial and final portions of the accelerogram. The objective of baseline correction is to diminish the long-period noise or drift introduced in the signal recording and playback process.

5. Integrate the baseline corrected velocity time history to obtain a displacement time history, assuming zero initial displacement. Differentiate the baseline corrected velocity time history to obtain the corrected acceleration time history.

A Matlab package has been written to fulfill this data processing procedure. The package is also able to plot acceleration, velocity, and displacement time histories, to plot horizontal displacement particle trajectories, and to compute and plot Response Spectra and Drift Demand Spectra [3]. There are a total of 15 Matlab routines (functions) in this package. Sample Matlab programs are contained in Appendix 1. SAMPLE1.M and SAMPLE2.M demonstrate the usage of the routines. Appendix 2 lists all routines. A floppy disk containing all routines is included with this report.

REFERENCES

1. Iwan, W.D., 'Near-Field Considerations in Specification of Seismic Design Motions for Structure,' Proceedings of the 10th European Conference on Earthquake Engineering, Vienna, Austria, August 28-September 2, 1994.

2. Iwan, W.D. and X.D. Chen, 'Important Near-Field Ground Motion Data from the Landers Earthquake,' Proceedings of the 10th European Conference on Earthquake Engineering, Vienna, Austria, August 28-September 2, 1994.

3. Iwan, W.D., 'Drift Demand Spectra for selected Northridge Sites,' EERL 95-07, Earthquake Engineering Research Laboratory, California Institute of Technology, Pasadena, December, 1995.

APPENDIX 1. EXAMPLES

```
% SAMPLE1
%
% This is a sample program used to demonstrate how to process a Near-
% Field strong motion record. In this example, the raw data of the
% east component of kobe station (KOB) in the 1995 Kobe earthquake
% is processed to obtain corrected acceleration, velocity, displace-
% ment, Fourier spectrum, response spectrum, and drift demand
% spectrum.
%
% Last Revised September 11, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

% 1. Input raw data file 'kobe' and instrument paramenters
%                sma  : 1 for SMA-1, 2 for SMA-2
%                dt   : Time Step
%                xi   : Damping Ratio
%                wnhz : Natural Frequency (Hz)
%                wchz : Cornor Frequency (Hz)
%                sen  : Sensitivity
load kob.e; rdata=kob(:,2); tit='KOB (East)';
sma=1; dt=0.02; wnhz=250; xi=0.7; wchz=0; sen=1;

% 2. Instrument correction
[racce,rvelo]=instrument(rdata,sma,dt,wnhz,xi,wchz,sen,tit);

% 3. Baseline correction
%       t1:     First breakpoint (second)
%       order1: Order of the first fitting curve
%       t2:     Second breakpoint (second)
%       order2: Order of the second fitting curve

t1=4.5; order1=0; t2=20; order2=1;
[acce,velo,disp,base]=baseline(t1,order1,t2,order2,dt,rvelo,tit);

% 4. Compute Fourier, response, and drift demand spectra

spectra(acce,velo,disp,dt,tit)
```

```
% SAMPLE2
%
% This is a sample program used to demonstrate how to find maximum
% velocity direction from 2 perpendicular directions and how to plot
% time histories and particle trajectory.
%
% Last Revised September 11, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

% 1. Input 2 perpendicular data file 'kobe' and 'kobn'
% dt    : Time Step
% xlimit: Length of Time Axis

load kobe; a=kobe; load kobn; b=kobn; dt=0.02; xlimit=40; tit='KOB ';
tit1=[tit '(East)']; tit2=[tit '(North)']; tit4=[tit 'Particle Trajecto
ry'];

% 2. Find maximum velocity
[alpha,dire,a1,v1,d1]=maxvelo(a(:,1),a(:,2),a(:,3),b(:,1),b(:,2),b(:,3)
);
tit3=[tit dire];

% 3. Save maximum velocity in file 'kobmv1'
kobmv1=[a1 v1 d1];
save kobmv1 kobmv1 -ascii

% 4. Plot time histories
avdplot(1,[1:1:length(a(:,1))]*dt,a(:,1),a(:,2),a(:,3),tit1,xlimit)
print plot1

avdplot(2,[1:1:length(b(:,1))]*dt,b(:,1),b(:,2),b(:,3),tit2,xlimit)
print plot2

avdplot(3,[1:1:length(a1(:,1))]*dt,a1,v1,d1,tit3,xlimit)
print plot3

% 5. Plot trajectory
traj(4,a(:,3),b(:,3),tit4,'East (cm)','North (cm)')

% 6. Set axis in trajectory plot
axis([-20 20 -10 30])
print plot4
```

APPENDIX 2. LISTINGS OF ROUTINES

```
% AVDPLOT
% AVDPLOT(fignumber,T,acce,velo,disp,tit,xlimit)
%
% Description:  Plot Acceleration, Velocity and Displacement
%
% Last Revised July 7, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function avdplot(fignumber,T,acce,velo,disp,tit,xlimit)

figure(fignumber); orient tall;

[amax]=rmax(acce); [vmax]=rmax(velo); [dmax]=rmax(disp);
amax=['Max = ',int2str(round(amax)),' cm/s/s'];
vmax=['Max = ',int2str(round(vmax)),' cm/s'];
dmax=['Max = ',int2str(round(dmax)),' cm'];

subplot(3,1,1); plot(T,acce,'LineWidth',1);
ylabel('Acceleration (cm/s/s)','FontWeight','bold');
title(tit,'FontSize',[15],'FontWeight','bold');
text('Position',[0.7 0.9],'String',amax,'Unit','normal');
set(gca,'Xlim',[0; xlimit]);

subplot(3,1,2); plot(T,velo,'LineWidth',1);
ylabel('Velocity (cm/s)','FontWeight','bold');
text('Position',[0.7 0.9],'String',vmax,'Units','normal')
set(gca,'Xlim',[0; xlimit]);

subplot(3,1,3); plot(T,disp,'LineWidth',1);
ylabel('Displacemnt (cm)','FontWeight','bold');
xlabel('Time (second)','FontWeight','bold');
text('Position',[0.7 0.9],'String',dmax,'Units','normal')
set(gca,'Xlim',[0; xlimit]);

set(gcf,'PaperPosition',[1,1.25,7,9]);

return
```

```
% BASELINE
% [acce,velo,disp,base]=baseline(t1,order1,t2,order2,dt,rvelo,tit)
%
% Description:  Second part of baseline correction routine. It performs
%               1. Polynomial fits to the 1st and 3rd parts of raw velo
city
%               2. Spline fit to the 2nd part of raw velocity
%               3. Baseline correction to get velocity
%               4. Obtain acceleration and displacement
%
% Input:        t1:     First breakpoint (second)
%               order1: Order of the first fitting curve
%               t2:     Second breakpoint (second)
%               order2: Order of the second fitting curve
%               rvelo:  Raw Velocity (cms/s)
%
%
% Output:       acce:   Corrected acceleration (cm/s/s)
%               velo:   Corrected velocity (cm/s)
%               disp:   Corrected Displacement (cm)
%               base:   Baseline (cm/s)
%
% Reference:    INSTRUMENT
%
% Last Revised June 20, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [acce,velo,disp,base]=baseline(t1,order1,t2,order2,dt,rvelo,ti
t)

np=length(rvelo); T=[1:1:np]'*dt; tlabel='Time (second)';

% 1. Polynomial fits to the 1st and 3rd parts of raw velocity
np1=round(t1/dt); np2=round(t2/dt); t1=np1*dt; t2=np2*dt;
T1=[dt:dt:t1]'; T3=[t2:dt:T(np)]';
p1=polyfit(T1,rvelo(1:np1),order1); f1=polyval(p1,T1);
p3=polyfit(T3,rvelo(np2:np),order2); f3=polyval(p3,T3);

% 2. Spline fit to the 2nd part of raw velocity
a=f1(np1); b=f3(1); dt12=12*dt; t21=t2-t1;
c=(3*f1(np1-4)-16*f1(np1-3)+36*f1(np1-2)-48*f1(np1-1)+25*a)/dt12;
d=(-25*b+48*f3(2)-36*f3(3)+16*f3(4)-3*f3(5))/dt12;

for i=np1+1:np2-1
 x=T(i); x20=x-t1; x21=x-t2; x202=x20*x20; x212=x21*x21;
 f2(i-np1,1)=(1+2*x20/t21)*x212*a+(1-2*x21/t21)*x202*b+x20*x212*c+x21*x
202*d;
end

% 3. Baseline correction to get corrected velocity
base=[f1; f2/t21/t21; f3]; velo=rvelo-base;
xyplot(4,T,rvelo,tit,tlabel,'Raw Velocity and Baseline (cm/s)');
hold on;
xyplot(4,T,base,tit,tlabel,'Raw Velocity and Baseline (cm/s)');
hold off
```

```
% 4. Obtain acceleration and displacement
acce=dif(velo,dt); disp=inte(velo,dt);
avdplot(5,T,acce,velo,disp,tit,length(acce)*dt)

avdbr=[acce velo disp base rvelo];
save avdbr.dat avdbr -ascii
save baseline.tm  t1 order1 t2 order2 -ascii
```

```
% DIF:
% [y]=dif(x,dt)
%
% Description:  Obtain vector y by differentiating vector x using centr
al
%               difference rule
%               dt: time step
%
% Last Revised June 20, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [y]=dif(x,dt)
np=length(x);
for i=2:np-1
 y(i)=x(i+1)-x(i-1);
end
y(1)=(x(2)-x(1))*2; y(np)=(x(np)-x(np-1))*2;
y=y'/(dt+dt);
return
```

```
% DRIFTPLOT
% driftplot(fignumber,tit,T,dir1,drift1,dir2,drift2,dir3,drift3)
%
% Description:  Plot Drift Demand Spectrum of a shear building
%
% Last Revised July 21, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function driftplot(fignumber,tit,T,dir1,drift1,dir2,drift2,dir3,drift3)

figure(fignumber); orient tall;
set(gcf,'PaperPosition',[1,1,7,9]);
tex='Damping Ratios = 0, 2, 5, 10 %';

dmax1=round(max(drift1(:,1))+0.5);
dmax2=round(max(drift2(:,1))+0.5);
dmax3=round(max(drift3(:,1))+0.5);

subplot(3,1,1); plot(T,drift1,'LineWidth',1);
tit1=['Drift Demand Spectrum (' tit ')']; title(tit1,'FontWeight','Bold
');
ylabel(dir1,'FontWeight','bold');
text('FontSize',[10],'Position',[0.6 0.92],'String',tex,'Unit','normal'
);
set(gca,'Xlim',[0 5.5],'XTick',[0:5],'Ylim',[0; dmax1]);

subplot(3,1,2); plot(T,drift2,'LineWidth',1);
ylabel(dir2,'FontWeight','bold');
text('FontSize',[10],'Position',[0.6 0.92],'String',tex,'Unit','normal'
);
set(gca,'Xlim',[0 5.5],'XTick',[0:5],'Ylim',[0; dmax2]);

subplot(3,1,3); plot(T,drift3,'LineWidth',1);
ylabel(dir3,'FontWeight','bold');
text('FontSize',[10],'Position',[0.6 0.92],'String',tex,'Unit','normal'
);
xlabel('Period (second)','FontWeight','bold');
set(gca,'Xlim',[0 5.5],'XTick',[0:5],'Ylim',[0; dmax3]);

return
```

```
% DRIFTSPEC
% [drift]=driftspec(ns,dt,velo,disp)
%
% Description:  Compute Drift Demand Spectrum of a shear building
%
% Input:        ns,dt:  Number of steps, time step
%               velo:   Corrected velocity (cm/s)
%               disp:   Corrected displacement (cm)
%
% Output:       PERIOD: Natural frequency of the shear building
%               drift:  Maximum Interstory Drift
%
% Last Revised July 21, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [PERIOD,drift]=driftspec(ns,dt,velo,disp)

xi=[0 0.02 0.05 0.1 0.2]; PERIOD=[0.2:0.2:5];

for k=1:length(xi)
k
        pixi=pi*xi(k);
  for j=1:length(PERIOD)
j
        T2=2/PERIOD(j); pixi2T=pixi*T2;
    for i=1:ns
        t=i*dt; N=floor(t*T2); sum=0;
      for l=1:N
        n=ceil((t-l/T2)/dt)+1;
        sum=sum+(-1)^l*exp(-l*pixi)*(velo(n)+pixi2T*disp(n));
      end
        driftmax(i)=velo(i)+pixi2T*disp(i)+sum+sum;
    end
      drift(j,k)=0.0093873*PERIOD(j)^(-1/3)*max(abs(driftmax));
  end
end

plot(PERIOD,drift);
xlabel('Period (second)')
ylabel('Interstory Drift Demand')

return
```

```
% INSTRUMENT
% [racce,rvelo]=instrument(rdata,sma,dt,wnhz,xi,wchz,sen,tit)
%
% Description:  First part of baseline correction routine. It performs
%               1. First order linear fit to raw input data
%               2. Instrument correction
%               3. Obtain raw velocity from raw acceleration by integra
tion
%
% Input:        rdata: Input Raw Data in a column
%               sma  : = 1 for SMA-1, = 2 for SMA-2
%               dt   : Time Step
%               xi   : Damping Ratio
%               wnhz : Natural Frequency (Hz)
%               wchz : Cornor Frequency (Hz)
%               sen  : Sensitivity
%
% Output:       racce: Raw Acceleration (cm/s/s)
%               rvelo: Raw Velocity (cm/s)
%
% Note:         Unevenly spaced data should be first interpolated to
%               evenly space data before using BASELINE1
%
% Reference:    BASELINE, INTERPOLATE
%
% Last Revised June 20, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [racce,rvelo]=instrument(rdata,sma,dt,wnhz,xi,wchz,sen,tit)

% Some constant
pi2=pi+pi; wn=pi2*wnhz; wn2=wn*wn; xiwn2=2*xi*wn; wc=pi2*wchz;
[np, dummy]=size(rdata); T=(1:1:np)'*dt;
tlabel='Time (second)';

% Plotting Raw Data
xyplot(1,T,rdata,tit,tlabel,'Raw Data');

% 1. First order linear fit to raw input data
racce=dtrend(rdata,1);

% 2. Instrument correction
dum1=dif(racce,dt); dum2=dif(dum1,dt);
dum=dum2+xiwn2*dum1+wn2*racce;
if sma == 1                                % SMA-1
 csma=wn2;
else                                       % SMA-2
 dum3=inte(racce,dt);
 dum=dum+wc*dum1+xiwn2*wc*acce+wc*wn2*dum3;
% 32609.173 = 4*pi*pi*826, -- obtained from testing of SMA-2 by X.D. Ch
en
 csma=32609.173;
end
racce=sen*dum/csma;
xyplot(2,T,racce,tit,tlabel,'Raw Acceleration (cms/s/s)')
```

```
% 3. Obtain raw velocity from raw acceleration by integration
rvelo=inte(racce,dt);
xyplot(3,T,rvelo,tit,tlabel,'Raw Velocity (cms/s)')
```

```
% INTE:
% [y]=inte(x,dt)
%
% Description:  Obtain vector y by integrating vector x using trapezoid
al rule
%               dt: time step
%
% Last Revised June 20, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [y]=inte(x,dt)
dt2=dt/2; y(1)=0;
for i=2:length(x)
 y(i)=y(i-1)+(x(i-1)+x(i))*dt2;
end
y=y';
return
```

```
% MAXVELO
% [alpha,dire,amax1,vmax1,dmax1,amax2,vmax2,dmax2]=maxvelo(a1,v1,d1,a2,
v2,d2)
%
% Description:  Rotate acceleration, velocity, and displacement to the
%               maximum velocity direction
%
% Input:        a1, v1, d1:     AVD in X direction
%               a2, v2, d2:     AVD in Y direction
%
% Output:       alpha, dire:    Maximum velocity direction measured fro
m X
%                               (degree, counter clockwise) and N00E fo
rmat
%               amax1 - dmax1:  AVD in maximum velocity direction
%               amax2 - dmax2:  AVD normal to maximum velocity directio
n
%
% Last Revised July 12, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [alpha,dire,amax1,vmax1,dmax1,amax2,vmax2,dmax2]=maxvelo(a1,v1
,d1,a2,v2,d2)

% Find Maximum velocity and direction angle alpha
maxv=v1.*v1+v2.*v2;
[maxvalue, i]=max(abs(maxv));
alpha=atan2(v2(i),v1(i))*180/pi;

% Rotate to the maximum velocity direction
[amax1, amax2]=rot(a1,a2,alpha);
[vmax1, vmax2]=rot(v1,v2,alpha);
[dmax1, dmax2]=rot(d1,d2,alpha);

% Transfer alpha in N00E format and save in dire
if alpha >= 0
    t1='N';
  if alpha <= 90
    t2='E'; a=90-alpha;
  else
    t2='W'; a=alpha-90;
  end
else
    t1='S'
  if alpha <= -90
    t2='W'; a=-90-alpha;
  else
    t2='E'; a=90+alpha;
  end
end
dire=['(' t1 int2str(round(a)) t2 ')'];

return
```

```
% [SD,PSV,PSA,SV,SA,ED]=RESPONSE(f,dt)
%
% Purpose:  Computes elastic response spectra for SDOF systems with per
iods
%           T and damping ratios xi subjected to a support acceleration
 f(t)
%           sampled at a step dt. Zero initial conditions are assumed.
%
%           ydd(t) + 2 xi w yd(t) + w^2 y(t) = -f(t)    y(0)=0, yd(0)=0

%
% Input:        f  = excitation vector
%               dt = sample time in f
%
% Output:       SD  = Relative displacement response spectrum
%               PSV = Pseudo-relative-velocity
%               PSA = Pseudo-absolute-acceleration
%               SA  = Absolute-acceleration
%               SV  = Relative-velocity
%               ED  = Spectra of energy dissipation per unit mass
%
% Last Revised June 30, 1996
% Copyright (c) 1996 by L.J. Wang
% ----------------------------------------------------------------

function [SD,PSV,PSA,SV,SA,ED]=response(f,dt)

% Define period and damping ratio as in CDMG standard report
T=[.040 .042 .044 .046 .048 .050 .055 .060 .065 .070 .075 .080 .085 .09
0 .095 .100 .110 .120 .130 .140 .150 .160 .170 .180 .190 .200 .220 .240
.260 .280 .300 .320 .340 .360 .380 .400 .420 .440 .460 .480 .500 .550 .6
00 .650 .700 .750 .800 .850 .900 .950 1.000 1.100 1.200 1.300 1.400 1.50
0 1.600 1.700 1.800 1.900 2.000 2.200 2.400 2.600 2.800 3.000 3.200 3.40
0 3.600 3.800 4.000 4.200 4.400 4.600 4.800 5.000 5.500 6.000 6.500 7.00
0 7.500 8.000 8.500 9.000 9.500 10.000 11.000 12.000 13.000 14.000 15.00
0 16.000 17.000 18.000 19.000 20.000 22.000 24.000 27.000 30.000];
xi=[0 0.02 0.05 0.1 0.2];

for j=1:length(xi)
  for i=1:length(T)
      w=2*pi/T(i); C=2*xi(j)*w; K=w*w; y(:,1)=[0; 0];
      A=[0 1; -K -C]; Ae=expm(A*dt); AeB=A\(Ae-eye(2))*[0; -1];
    for k=2:length(f)
      y(:,k)=Ae*y(:,k-1)+AeB*f(k);
    end
      SD(i,j)=max(abs(y(1,:)));
      PSV(i,j)=SD(i,j)*w;
      PSA(i,j)=SD(i,j)*w*w;
      SV(i,j)=max(abs(y(2,:)));
      SA(i,j)=max(abs([K*y(1,:)+C*y(2,:)]));
      ED(i,j)=C*dt*sum(y(2,:).^2);
  end
end
return
```

```
% RMAX
% [xmax]=rmax(x)
%
% Description:  Find absolute maximum for vector x and save the relativ
e
%               value of maximum in xmax
%               For example, rmax([10,-18,3])=-18, rmax([-13,20,0])=20
%
% Last Revised July 7, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [xmax]=rmax(x)

if abs(max(x)) > abs(min(x))
xmax=max(x);
else
xmax=min(x);
end

return
```

```
% ROT
% [x2,y2]=rot(x1,y1,alpha)
%
% Description:  Rotate vectors x1 and y1 by degree alpha (counter-
%               closewise) to vectors x2 and y2
%
% Last Revised July 7, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function [x2,y2]=rot(x1,y1,alpha)

alpha=alpha*pi/180;
x2= x1*cos(alpha)+y1*sin(alpha);
y2=-x1*sin(alpha)+y1*cos(alpha);

return
```

```
% SPECTRA
% spectra(acce,velo,disp,dt,tit)
%
% Description:  Third part of baseline correction routine. It performs
%               1. Calculation and ploting of Fourier Spectrum
%               2. Calculation and ploting of Response Spectrum
%               3. Calculation and ploting of Drift spectrum
%
% Input:        acce:   Corrected acceleration (cm/s/s)
%               velo:   Corrected velocity (cm/s)
%               disp:   Corrected displacement (cm)
%               base:   Baseline (cm/s)
%               rvelo:  Raw velocity (cms/s)
%
% Output:       Magfa:  Fourier amplitude spectrum of acceleration
%               Phasefa:Fourier phase spectrum of acceleration
%               SD:     Relative displacement response spectrum
%               PSV:    Pseudo-relative-velocity
%               PSA:    Pseudo-absolute-acceleration
%               SA:     Absolute-acceleration
%               SV:     Relative-velocity
%               ED:     Spectra of energy dissipation per unit mass
%
% Reference:    INSTRUMENT, BASELINE
%
% Last Revised July 7, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function spectra(acce,velo,disp,dt,tit)

% 1. Calculation and ploting of Fourier Spectrum

l2=length(acce)/2; freq=[1:1:l2]/l2/2/dt;
fa=fft(acce); Magfa=abs(fa(1:l2)); Phasefa=angle(fa(1:l2))*180/pi;

figure(6); orient tall;

subplot(2,1,1); loglog(freq,Magfa,'LineWidth',1); grid on;
ylabel('Acceleration (cm/s/s)','FontSize',[15],'FontWeight','bold');
title(tit,'FontSize',[15],'FontWeight','bold');

subplot(2,1,2); semilogx(freq,Phasefa,'LineWidth',1); grid on;
xlabel('Frequency (Hz)','FontSize',[15],'FontWeight','bold');
ylabel('Phase (degree)','FontSize',[15],'FontWeight','bold');

% 2. Calculation and ploting of Response Spectrum
T=[.040 .042 .044 .046 .048 .050 .055 .060 .065 .070 .075 .080 .085 .09
0 .095 .100 .110 .120 .130 .140 .150 .160 .170 .180 .190 .200 .220 .240
.260 .280 .300 .320 .340 .360 .380 .400 .420 .440 .460 .480 .500 .550 .6
00 .650 .700 .750 .800 .850 .900 .950 1.000 1.100 1.200 1.300 1.400 1.50
0 1.600 1.700 1.800 1.900 2.000 2.200 2.400 2.600 2.800 3.000 3.200 3.40
0 3.600 3.800 4.000 4.200 4.400 4.600 4.800 5.000 5.500 6.000 6.500 7.00
0 7.500 8.000 8.500 9.000 9.500 10.000 11.000 12.000 13.000 14.000 15.00
0 16.000 17.000 18.000 19.000 20.000 22.000 24.000 27.000 30.000];
```

```
[SD,PSV,PSA]=response(acce,dt);

tripart(7,PSV,tit);

% 3. Calculation and ploting of Drift Spectrum
[PERIOD,drift]=driftspec(ns,dt,velo,disp);
```

```
% TRAJ
% TRAJ(fignumber,x,y,tit,xlab,ylab)
%
% Description:  Plot particle trajectory of vectors x and y
%
% Last Revised July 7, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function traj(fignumber,x,y,tit,xlab,ylab)

figure(fignumber);
plot(x,y,'LineWidth',1);
axis square; axis equal;
title(tit,'FontSize',[15],'FontWeight','bold');
xlabel(xlab,'FontSize',[15],'FontWeight','bold');
ylabel(ylab,'FontSize',[15],'FontWeight','bold');
set(gcf,'PaperPosition',[1,2,7,7]);

return
```

```
% TRIPART:
% function tripart(fignumber,PSV,tit)
%
% Description:  Tripartite plot of Response Spectrum
%
% Last Revised July 1, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function tripart(fignumber,PSV,tit)

T=[.040 .042 .044 .046 .048 .050 .055 .060 .065 .070 .075 .080 .085 .09
0 .095 .100 .110 .120 .130 .140 .150 .160 .170 .180 .190 .200 .220 .240
.260 .280 .300 .320 .340 .360 .380 .400 .420 .440 .460 .480 .500 .550 .6
00 .650 .700 .750 .800 .850 .900 .950 1.000 1.100 1.200 1.300 1.400 1.50
0 1.600 1.700 1.800 1.900 2.000 2.200 2.400 2.600 2.800 3.000 3.200 3.40
0 3.600 3.800 4.000 4.200 4.400 4.600 4.800 5.000 5.500 6.000 6.500 7.00
0 7.500 8.000 8.500 9.000 9.500 10.000 11.000 12.000 13.000 14.000 15.00
0 16.000 17.000 18.000 19.000 20.000 22.000 24.000 27.000 30.000];

figure(fignumber); orient tall; clf; loglog(T,PSV,'LineWidth',1.5);
axis([0.01 100 0.1 1000]); axis square; set(gca,'TickLength',[.02 .02])
;
xbox=[0.01 0.01 0.01 100; 100 100 0.01 100];
ybox=[0.1 1000 0.1 0.1; 0.1 1000 1000 1000];
line(xbox,ybox,'LineWidth',1.5);

title(tit,'FontSize',[15],'FontWeight','bold');
xlabel('Period (second)','FontSize',[15],'FontWeight','bold');
ylabel('PSV (cm/second)','FontSize',[15],'FontWeight','bold');

text('FontSize',[15],'FontWeight','bold','pos',[.043 224],'str','PSA (g
)','rot',[+45]);
text('FontSize',[15],'FontWeight','bold','pos',[14.5 150],'str','SD (cm
)','rot',[-45]);
text('FontSize',[15],'pos',[1 0.15],'str','Damping Ratios = 0, 2, 5, 10
, 20 %', 'HorizontalAlign','Center');

pi2=pi+pi;
for i=1:3
  x=pi2*10^(i-3); y1=x*10; y2=10^(-i+4)/pi2;
  line([x; 0.01],[ 0.1; y1],'LineStyle',':');
  line([x; 100], [ 0.1; y2],'LineStyle',':');
  line([x; 0.01],[1000; y2],'LineStyle',':');
  line([x; 100], [1000; y1],'LineStyle',':');
end
  x=pi2*10; y1=x*10; y2=10^(-log10(pi2));
  line([x; 0.01],[ 0.1; y1],'LineStyle','-');
  line([x; 0.01],[1000; y2],'LineStyle','-');

for i=1:4
    x1=pi2*10^(i-3); x2=x1*10^(-1/2); y1=10^(i-1); y2=y1*10^(-1/2);
    z1=10^(-i+3); z2=z1*10^(1/2);
    l1=1.15; l2=1.20; l3=1.13; l4=1.28; l5=1.56; l6=1.19; l7=1.42; l8=1
.08;
    s1=int2str(2*i-4); s2=int2str(2*i-5);
```

```
    s3=int2str(-2*i+3); s4=int2str(-2*i+4);
  if i ~= 4
      text('pos',[x1/11 y1/12],'str','10', 'rot',[-45]);
      text('pos',[x1/15 z1/16],'str','10', 'rot',[+45]);
      text('FontSize',[8],'pos',[x1*13 y1/14],'str',s1, 'rot',[-45]);
      text('FontSize',[8],'pos',[x1/17 z1*18],'str',s3, 'rot',[+45]);
    if i ~= 1
      text('pos',[x2/11 y2/12],'str','10', 'rot',[-45]);
      text('pos',[x2/15 z2/16],'str','10', 'rot',[+45]);
      text('FontSize',[8],'pos',[x2*13 y2/14],'str',s2, 'rot',[-45]);
      text('FontSize',[8],'pos',[x2/17 z2*18],'str',s4, 'rot',[+45]);
    end
  end
  for j=1:9
    19=1.07; if j==1; 19=1.13; end
    x11=x2*j^(1/2); x12=x11/19; y11=y2*j^(1/2); y12=y11*19;
    x21=x1*j^(1/2); x22=x21/19; y21=y1*j^(1/2); y22=y21*19;
    x31=x1*j^(-1/2); x32=x31*19; z11=z1*j^(1/2); z12=z11*19;
    x41=x2*j^(-1/2); x42=x41*19; z21=z2*j^(1/2); z22=z21*19;
    line([x11; x12],[y11; y12],'LineStyle','-');
    line([x21; x22],[y21; y22],'LineStyle','-');
    line([x31; x32],[z11; z12],'LineStyle','-');
    line([x41; x42],[z21; z22],'LineStyle','-');
  end
end
```

```
% XYPLOT:
% xyplot(fignumber,x,y,tit,xlab,ylab)
%
% Description:  Plot x-y with title, xlab, ylab on Figure fignumber
%
% Last Revised June 20, 1996
% Copyright (c) 1996 by L.J. Wang
% ------------------------------------------------------------------

function xyplot(fignumber,x,y,tit,xlab,ylab)

figure(fignumber);
plot(x,y,'LineWidth',1.5);
title(tit,'FontSize',[15],'FontWeight','bold');
xlabel(xlab,'FontSize',[15],'FontWeight','bold');
ylabel(ylab,'FontSize',[15],'FontWeight','bold');
return
```